附件 3

中國文化大學教師教學創新暨教材研發獎勵期末成果報告書

壹、計畫名稱

英文計畫名稱: Fundamental Design of Combinational and Sequential Logic Circuit

中文計畫名稱: 組合電路與序列電路基礎設計

貳、實施課程、授課教師姓名

課程名稱 : 邏輯設計 (Logic Desgin)

開課系級：電機工程學系 一 年級

授課教師：逄霖生

電子郵件 : pls@faculty.pccu.edu.tw

參、前言

組合電路與序列電路是數位電路的基礎，台灣在世界的工業鏈中，在數位設計的能力有目共睹，這個領域需要的相關科技人才，必須有組合電路與序列電路的知識。電機系在此專業領域特別重視相關的技術與知識。唯有組合電路與序列電路的基本技能十分紮實，才能有足夠的能力面對更大的挑戰。本教材創新計畫針對基本邏輯知識詳加介紹：包含Boolean logic(布林邏輯)，數位資料表示與實際基本邏輯閘的介紹應用。進階的數位序列電路的原理，相關的實際電路應用：包含正反器、閘極開關、計算器與Timer的設計與實際應用的電路。本教材將使用

Altera公司發展的EDA與FPGA硬體，提供學有實際動手操作的設備，希望透過專案設計的方式，提供學員了解硬體與軟體編寫及燒錄開發設計的過程，讓學生了解邏輯設計中數位組合電路與序列電路的設計。

肆、計畫特色及具體內容

邏輯電路設計與應用是台灣工業的強項，這個領域需要大量的3C相關科技人才。電機工程系的學習領域，包含了硬體實作與軟體應用，因此特別適合同學在此項目中發揮所長。但是數位邏輯電路應用基本技能包含邏輯能力的分析，組合與實際電路應用，如此才能有足夠的知識挑戰不同的問題。學生也應不斷地吸收新的知識，來面對計算機工程或數位電路設計日新月異的發展。本教材創新計畫針對基本邏輯知識與進階的數位序列電路的原理詳加介紹：內容有布林邏輯、數位資料、基本邏輯閘、數位應用電路、正反器、閘極開關、計算器與Timer的設計與實際應用的電路。本教材將使用Altera公司發展的EDA與FPGA硬體，提供學有實際動手操作的設備，讓學員有機會了解硬體與軟體編寫及燒錄開發設計的過程，加強學生對邏輯設計中組合電路與序列電路的瞭解。

Combinational Logic Circuit 數位組合電路

- Number Systems, Arithmetic, and codes
  - Digital Systems vs. Analog Systems

Digital Systems vs. Analog Systems

- Digital System
  - Information is represented and processed by a finite number of discrete digits.
  - Example: Binary Strings of 1's and 0's are used to represent information.
- Analog System
  - Information is represented and processed along a continuum.
  - Example: Consider making measurements with a ruler.

Number Systems

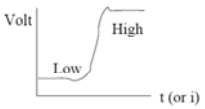
Positional Number Systems

- The above is a general form of a power series in radix *r*. A number *N* expressed in base-*r* system has coefficients multiplied by powers of *r*:
- $N = d_{n-1} * r^{n-1} + d_{n-2} * r^{n-2} + \ldots d_1 * r^1 + d_0 * r^0 + d_{-1} * r^{-1} + d_{-2} * r^{-2} + \ldots d_{-m} * r^{-m}$

Binary Arithmetic

Binary Number

- Two discrete values are used in digital systems only.
- The values of a binary number could be
  - False/True
  - Low/High
  - 0/1
  - Yes/No
  - Go/No Go



Ternary Arithmetic

Ternary Number

Similar to the binary number, but the number of ternary number includes {0, 1, 2}. There are total three digital numbers to represent ternary system.

- Binary and Decimal Number Base Conversions

Decimal to Binary Conversion

- A decimal number *N*, indicating as "i.f" (i.e. 3.13159), includes two parts: the first part is integer part, *i*, and the second part is fractional part, *f*.
- The procedures of decimal number to binary conversion are shown as follows:
  1. Convert integer part
  2. Convert fractional part
  3. Combine integer and fractional parts together

Polynomial Method of Number Conversion (Givone Text Book)

- The number $N_{(r1)}$ of Base-r1
- $N_{(r1)} = (d_{n-1}d_{n-2} \cdots d_1d_0.\, d_{-1} \cdots d_{-m})_{(r1)}$, $0 <= d_i <= (r1-1)$
- $N_{(r1)} = d_{n-1(r1)} * r_1^{n-1}{}_{(r1)} + d_{n-2(r1)} * r_1^{n-2}{}_{(r1)} + \cdots + d_{0(r1)} * r_1^{0}{}_{(r1)} + d_{-1(r1)} * r_1^{-1}{}_{(r1)} + \cdots + d_{m(r1)} * r_1^{-m}{}_{(r1)}$
- $= d_{n-1(r1)} * 10^{n-1}{}_{(r1)} + d_{n-2(r1)} * 10^{n-2}{}_{(r1)} + \cdots + d_{0(r1)} * 10^{0}{}_{(r1)} + d_{-1(r1)} * 10^{-1}{}_{(r1)} + \cdots + d_{-m(r1)} * 10^{-m}{}_{(r1)}$

  Note the quantity of 2 in binary is represent by $10_{(2)}$

  Note the quantity of 3 in ternary is represent by $10_{(3)}$
- The number $N_{(r2)}$ of Base-r2
- $N_{(r2)} = d_{n-1(r2)} * r_1^{n-1}{}_{(r2)} + d_{n-2(r2)} * r_1^{n-2}{}_{(r2)} + \ldots + d_{m(r2)} * r_1^{-m}{}_{(r2)}$
- $N_{(r1)}$ is converted into of $N_{(r2)}$

- Iterative method of number conversion

Iterative method of converting integer (Givone Text Book)

- To convert an integer in base r_1 into its equivalent integer in r_2.
- Divide the number by r_2, then the remainder is the 0-th order digit.
- Repeat the previous step to get 1st order digit,
- Repeat the division process of r_2 until the remainder is zero.

- Signed and Unsigned Numbers

Signed Numbers

- Signed numbers denote whether the magnitudes is positive or negative.
- The form to show a singed number is called as “sign-magnitude representation”.
- The sign-magnitude representation includes the following methods:
  - Proceeding with signed symbol, (+) or (-). (used in regular number representation)
  - In digital system, it utilizes the binary digit 0 to denote the plus sign and the binary digit 1 to denote the minus sign.

■ Complements of Number

Complement notation

- The complement notation uses the most significant bit represents the sign bit, indicating whether the number is positive or negative.

■ Overflow in unsigned and signed number

Overflow

- the binary number operations need *n* + 1 digit to do *n* digit number operations.

■ Floating number (IEEE-754 standard)

5-4 浮點數的表示法

- 浮點數表示法
  - 不同 CPU 雖有其各自的浮點數表示法, 但一般較常採用 IEEE 協會所訂定的浮點數表示法標準(IEEE Standa rd 754), 其分成單精確度 (Single Precision) 及雙精確度 (Double Precision) 二種。

    兩者的差別在於單精確度是以 32 個位元來表示浮點數, 雙精確度則是用 64 位元表示。

50

■ Binary Codes (BCD Codes)

Decimal Codes

- BCD
  - Binary Coded Decimal•
- Represents decimal digits
  - 0 → 9
- It will need 4 binary digits to represent ten numbers, but leaves 6 combinations un-used.
- Weighted Codes
  - Positional of number indicates weight ($w_3w_2w_1w_0$)
- A number $N = w_3.b_3+w_2.b_2+w_1.b_1+w_0.b_0$
- BCD codes include several different forms, such as 8421, 2421, 5421, 7536, 5043210 codes.

■ Unit-Distance Codes

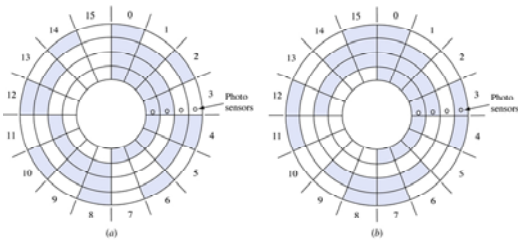
Unit-Distance Codes

- Only a single bit changes between any two successive coded integers
- The most popular of the unit distance codes are the Gray codes (named after their inventor Frank Gray).

表 3: Non-weighted decimal codes

Decimal Number	Gray Code
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

Angular position encoders. (*a*) Conventional binary encoder. (*b*) Gray code encoder.



■ ASCII & Unicode

American Standard Code for Information Interchange (ASCII)

$B_4B_3B_2B_1$	$B_7B_6B_5$ = 000	001	010	011	100	101	110	111
0000	NULL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB		7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

- Error-Detection Codes (Parity Code)

Error-Detection Codes

- A code is said to be n-error detecting if the minimum of *n* errors that cannot be detected is *n* + 1
  - Error defined as a bit being complemented erroneously
- Distance between two code groups
  - The number of bits that must change so that the first code group becomes the second
- Minimum Distance
  - Minimum distance between any two valid code groups in a coding scheme
- Maximum number of detectable errors
  - $D = M - 1$
  - D is the error detecting capability of code
  - M minimum distance

- Error Correction Codes (Hamming Code)

Hamming Code

- Derived by R.W. Hamming
- Consider the case of four information bits
- Three parity bits are included
- Each calculated over a specified set of bits
- Let p_i represent parity bit i
- Let b_i represent parity information bit i
- 7 6 5 4 3 2 1 : Positions
- $b_4b_3b_2p_3b_1p_2p_1$: Code Group Format
- p_1 :: Even parity over positions 1, 3, 5, 7
- p_2 :: Even parity over positions 2, 3, 6, 7
- p_3 :: Even parity over positions 4, 5, 6, 7
- **Hamming code can detect and correct an error bit.**

BOOLEAN ALGEBRA AND COMBINATIONAL NETWORKS

Introduction to Boolean Algebra

Intro. to Boolean Algebra

- An algebra for symbolically representing problems in logic and analyzing them ***mathematically.***
- Based on work of George Boole, an English mathematician,
  - Published "An Investigation of the Laws of Thought",
  - Published in 1854
- Claude E. Shannon, Massachusetts Institute of Technology,
  - Showed how a Boolean algebra could be applied to certain engineering problems.
- Switching Circuit Theory
  - The study of Boolean algebra applied to logic design.
  - For any logic system consisting of elements with two-valued characteristics.
  - To describe terminal properties of a logic network.
  - To manipulation the network realization.
  - To simplification the network realization.

3

Intro. to Boolean Algebra

- The logic network described by Boolean algebra is divided into two categories:
  1. Combinational networks
     - The outputs of a logic network are a function of current inputs at any instant.
  2. Sequential networks
     - The outputs of a logic network are not only a function of current inputs but, in addition, depend upon the past history on inputs, i.e. the states of variables stored in memory.

4

Theorems & Postulates of Boolean Algebra

Postulates of Boolean Algebra

- P1: Operations (+) and (·) are closed
  $x + y \in B$ (6)
  $x \cdot y \in B$
- P2: Identity Elements
  - Identity elements exist, such that for every element $x \in B$
    $0 + x = x + 0 = x$ (9)
    $1 \cdot x = x \cdot 1 = x$ (Duality)
- P3: Commutative Law
  $x + y = y + x$ (12)
  $x \cdot y = y \cdot x$ (Duality)

Closure:
If X S and Y S then X.Y S
If X S and Y S then X+Y S

1

Postulates of Boolean Algebra

- P1: Operations (+) and (·) are closed
  $x + y \in B$ (6)
  $x \cdot y \in B$
- P2: Identity Elements
  - Identity elements exist, such that for every element $x \in B$
    $0 + x = x + 0 = x$ (9)
    $1 \cdot x = x \cdot 1 = x$ (Duality)
- P3: Commutative Law
  $x + y = y + x$ (12)
  $x \cdot y = y \cdot x$ (Duality)

Closure:
If X S and Y S then X.Y S
If X S and Y S then X+Y S

1

Two-Valued Boolean Algebra

Two-Valued Boolean Algebra

- The objective of this topic is to establish a relationship between a Boolean algebra and logic networks and how to show a Boolean algebra is a useful tool in logic network analysis and design.
- The two-valued Boolean algebra is also known as the switching algebra.
  - The analysis and design of logic network, it is common to refer to the two-valued Boolean algebra simply as a Boolean algebra without additional qualification.

35

■ Boolean Formulas and Functions

Boolean Formulas and Functions

- Boolean expression or formulas are constructed by connecting the Boolean algebra constants and variables with the Boolean operations.
- Boolean expressions are used to describe Boolean functions.
  - Example:
    Boolean expression : (x' + y) z
    Boolean function : f(x, y, z) = (x' + y) z
- The value of Boolean function is determined by any set of values of variables.

38

■ The Truth Table

The Truth Table

- A table listing the outputs for every possible combination of inputs for an n-input function
- Inputs
  - Enumerated on left
  - Count from [0…0] (all zeros) to [1…1] (all ones) in binary to enumerate all values
- Outputs
  - Enumerated on right
- Columns
  - *n* + 1 (minimum)
  - Often intermediate values are listed instead of just the output of the function
- Rows
  - 2^n

40

Truth table for the function f=(x'+y)z

x	y	z	x'	x' + y	f = (x' + y)z
0	0	0	1	1	0
0	0	1	1	1	1
0	1	0	1	1	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	1	0
1	1	1	0	1	1

41

■ Canonical Formulas of Boolean Algebra & Manipulations of Boolean Formulas

Canonical Formulas

- 2 types of expressions are obtained directly from a truth table
  - Minterm canonical formula
    - Special case of disjunctive normal formula
    - 2^n minterms exist for n Boolean variables
  - Maxterm canonical formula
    - Special case of conjunctive normal formula
    - 2^n maxterms exist for n Boolean variables

6

Manipulations of Boolean Formulas

- A Boolean function may be represented in different forms, with the same Boolean functions, by applying the postulates and theorems of a Boolean algebra.

19

■ Canonical Forms

附件 3

Canonical Forms

- The standard form of an equation consists of product or sum terms
  - Referred to as the canonical form
  - Two forms
  - POS
  - SOP

29

Minterm Canonical Formulas

- The significance of the minterm canonical formula is to uniquely describe a complete Boolean function.
- It can transfer any Boolean function and expression into a minterm canonical formula.

30

Maxterm Canonical Formulas

- The maxterm canonical formula can be uniquely described a complete Boolean function.
- It can transfer any Boolean function and expression into a minterm canonical formula.

32

■ Gates and Combinational Networks

Logic Networks

- Logic elements:
  - Gates: electronic circuits whose terminal properties corresponding to various Boolean operations.
  - Flip-flops: memory devices which can store logic constant.
- Logic diagram: the drawing of logic elements.
- Logic networks:
  - The interconnections of gates and flip-flops are formed the logic networks.
  - It is also called a *realization* (or implementation) of a Boolean algebra.

1

Gates

- Electronic circuits have only two possible steay-state voltage signal values appear at the terminals.
  - and-gate:
    - perform Boolean AND operation
  - or-gate:
    - perform Boolean OR operation
  - not-gate:
    - perform Boolean NOT operation
- Each gate has its specified logic diagram.

4

■ Incomplete Boolean Functions and Don’t-Care Conditions

附件 3

Incomplete Boolean Functions and Don't-Care Conditions

- Complete Boolean function: All the combinations of n input variables of a Boolean expression have uniquely functional value.
- Incomplete Boolean function: Not every combinations have been assigned a value to the output, marked with the symbol "x".

An example of a three-variable incomplete Boolean function and its complement

x	y	z	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	x
1	0	0	0
1	0	1	x
1	1	0	0
1	1	1	1

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	x
1	0	0	1
1	0	1	x
1	1	0	1
1	1	1	0

16

■ Additional Boolean Operations and Gates

Additional Boolean Operations and Gates

- NAND gates
- NOR gates
- Exclusive-OR gates

21

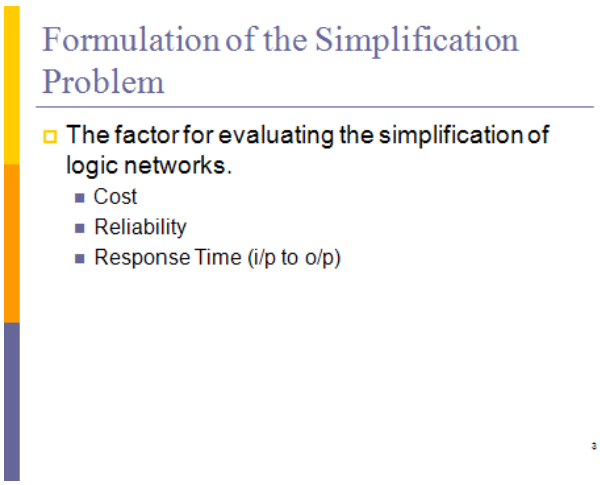
■ Physical Properties of Logic Gates

Gate Properties

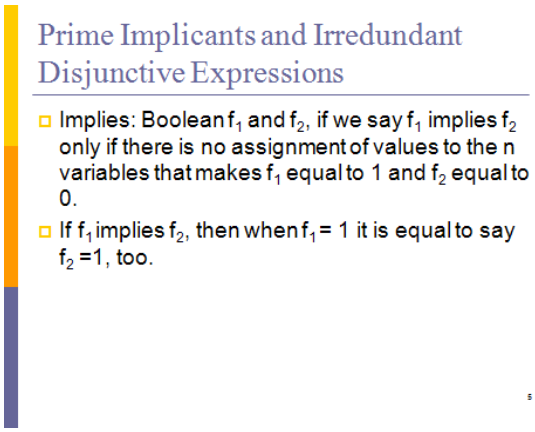
- Logic families
  - TTL (transistor-transistor logic)
  - ECL (emitter-coupled logic)
  - CMOS (complementary metal-oxide-semiconductor logic)

47

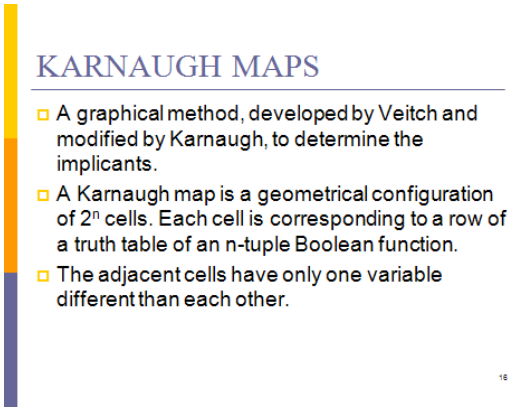
● Simplification of Boolean Expressions



- Prime Implicates and Irredundant Conjunctive Expressions



- KARNAUGH MAPS



- Using Karnaugh Maps to Obtain Minimal Expressions for Complete Boolean Functions

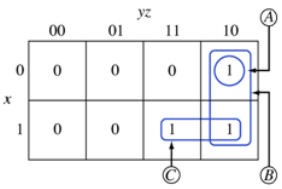
- By using the Karnaugh map, a Boolean expression can be minimized by eliminating unnecessary (redundant) subcubes.

Prime Implicants and Karnaugh Maps

- Assume a set of subcubes have been selected all 1-cells and no 0-cells being picked.
  - These corresponding product terms with respect to selected subcubes are implicants.
- Questions:
  How can one select the prime implicants from these subcubes.

  We are going to demonstrate by using an example.

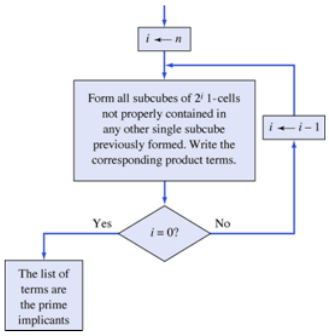
- Give a Karnaugh map, shown in the slide
  - The subcube A is an implicant, x'yz'.
  - The subcube A clearly can be included in subcube B, whose logic expression is yz'.
  - Since A is included in B (A subsumes B), the subcube is not a prime implicant.
  - There is no other bigger subcubes can include subcube B.
  - The yz' is a prime implicant.



Algorithm to find all prime implicants.

Figure 4.14

The general procedures for determining the prime implicants of a Boolean function is shown in flow chart:



■ Minimal Expression of Incomplete Boolean Functions

Minimal Sums (for incomplete Boolean function)

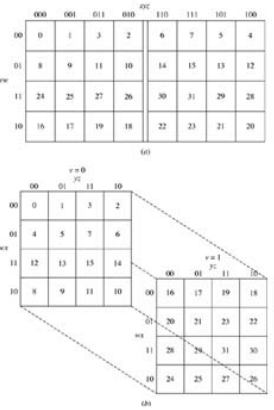
- For an incomplete Boolean function, the don't-care cells can be either treated as 1 or 0.
  - The rule of thumb is trying to use don't-care cells to get a larger subcube as possible.
  - But get as fewer subcubes as possible.
- Then proceed the Karnaugh map as a complete Boolean function to get its SOP form.

Note that it may exist more than one solutions.

■ Five-variable and Six-variable Karnaugh Maps

Five-variable Maps

- Five-variable Karnaugh maps combines two four-variable Karnaught maps together.
- The first four input variables are the same for each map, but the fifth one represents the differences of two maps.

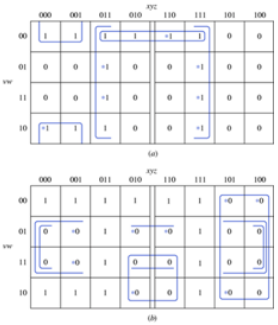


Map for $f(v,w,x,y,z) = \Sigma m(0, 1, 2, 3, 6, 7, 11, 15, 16, 17, 19, 23, 27, 31)$. (*a*) Subcubes for the minimal sum. (*b*) Subcubes for the minimal product.

Figure 4.28

$f(v,w,x,y,z) = w'x'y' + yz + v'w'y$

$f(v,w,x,y,z) = (w'+y)(w'+z)(v'+y'+z)(x'+y)$



- The Quine-McCluskey Method of Generating Prime Implicants and Prime Implicates

Prime Implicants and the Quine-McCluskey Method

- Quine-McCluskey method can use the minterms to get the minimal expressions
  - AB + A'B = B
- Each product term can be also represented by 0-1-dash notation, where 1 is used to denote an uncomplemented variable, 0 is used to denote complemented variable, and a dash for the absence of a variable.
- If there exists don't-care conditions, it will use 1 to replace it.

106

- Decimal Method for Obtaining Prime Implicants

Algorithm for Generating Prime Implicants

- Algorithm for generating prime implicants
  1. Express each minterm of the function in its binary representation.
  2. List the minterms by increasing index.
  3. Separate the sets of minterms of equal index with lines.
  4. Let i=0.
  5. Combine each term of index I with each term of index i+1. For each pair of terms that can combine, i.e., differ in exactly on bit position, place the newly formed term (also in 0-1-dash notation) in the section of index I of a new list, unless it is already present. In either event, place a check mark next to the two terms that combined (if not already checked). In the comparison process, a checked term does not disqualify it from further comparisons. After all pairs of terms with indices I and i+1 are inspected in the original list, a line is drawn under the last term in the new list.

To be continued

6. Increase I by 1 and repeat Step 5, the increase of I is continued until all terms are compared. The new list contains all those implicants of the function that have one less variable than those implicants in the generating list.
7. Each section of the new list formed has terms of equal index. Steps 4, 5 and t are repeated on this list to form another list. Recall that two terms combine only if they have their dashes in the same relatively positions and if they differ in exactly one bit position.
8. The process terminates when no new list is formed.
9. All terms without check mark are prime implicant.

121

- LOGIC DESIGN WITH MSI COMPONENTS AND PROGRAMMABLE LOGIC DEVICES
  - Binary Adders and Subtracter

Binary full adder

- Truth table for adder
- $S_i = x_i' y_i' c_i + x_i' y_i c_i' + x_i y_i' c_i' + x_i y_i c_i$
  $= (x_i \oplus y_i \oplus c_i)$
- $C_{i+1} = (x_i y_i + x_i c_i + y_i c_i)$

x	y	c_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

5

Binary subtracter

- Binary full subtracter
- $d_i = (x_i \oplus y_i \oplus b_i)$
- $b_{i+1} = (x'_i y_i + x'_i b_i + y_i b_i)$

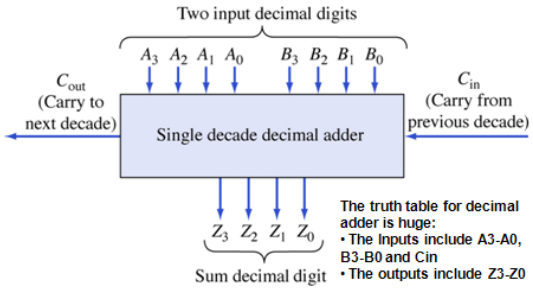
x	y	b_i	b_{i+1}	d_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

9

■ Decimal Adders & Comparators

Organization of a single-decade decimal adder

Figure 5.11

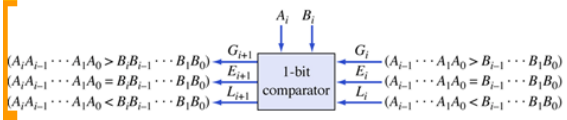


The truth table for decimal adder is huge:
- The Inputs include A3-A0, B3-B0 and Cin
- The outputs include Z3-Z0

25

Organization of a 1-bit comparator

Figure 5.15



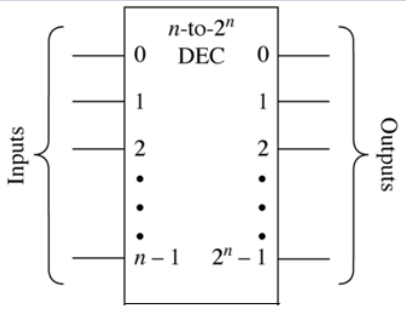
There are 5 inputs and 3 outputs for 1-bit comparator.

32

■ Decoders & Encoders

An *n*-to-2^n-line decoder symbol.

Figure 5.17

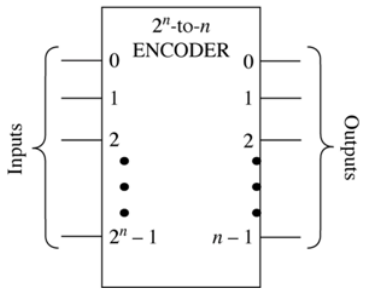


This n-to-2^n-line decoder is also called as "minterm generator.

4

A 2^n-to-n-line encoder symbol.

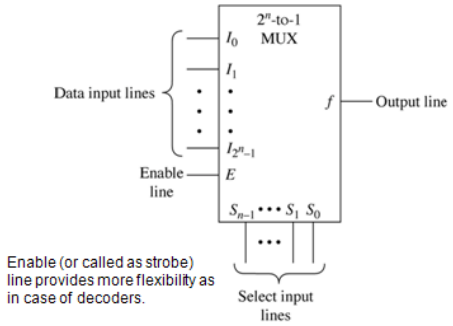
Figure 5.30



24

■ Multiplexers

A 2^n-to-1-line multiplexer symbol.
Figure 5.32



29

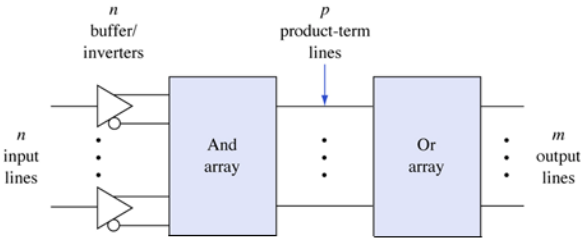
- Programmable Components & Programmable Logic Devices (PLDs)

Programmable logic devices

- Use large-scale integration technology to implement a large circuits onto a single chip
- Programmable read-only memory (PROM)
- Programmable logic array (PLA)
- Programmable array logic (PAL)
- PAL is the trademark for Advanced Micro Devices Inc.

91

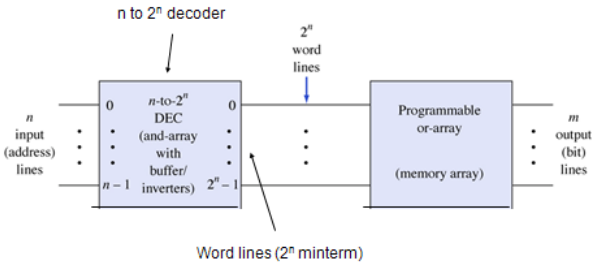
General structure of PLDs.
Figure 5.48



92

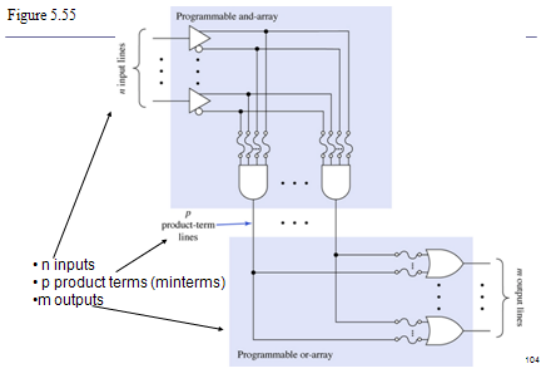
- Programmable Read-Only Memory (PROMs) & Programmable Logic Arrays (PLAs)

Structure of a PROM.
Figure 5.52



99

Logic diagram of an $n \times p \times m$ PLA.
Figure 5.55

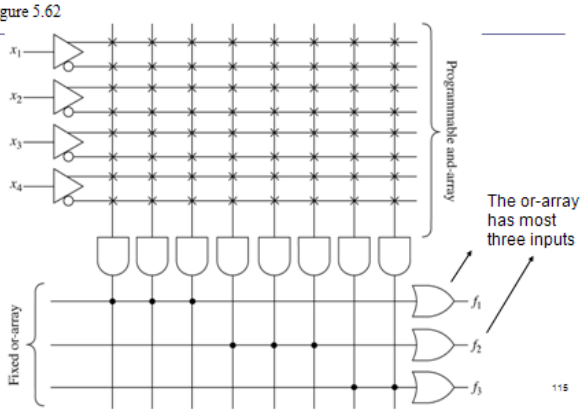


104

- Programmable Array Logic (PAL) Devices

A simple four-input, three-output PAL device.

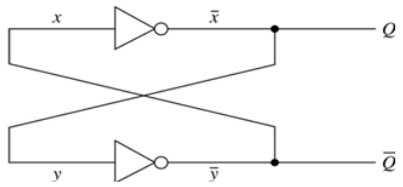
Figure 5.62



Sequential Logic Circuit 數位序列電路

- Flip-Flops and Simple Flip-Flop Applications
  - The Basic Bistable Element

- Bistable (FF)
  - If FF store 1 (x=1, y=0), its state or content is 1
  - If FF store 1 (x=0, y=1), its state or content is 0

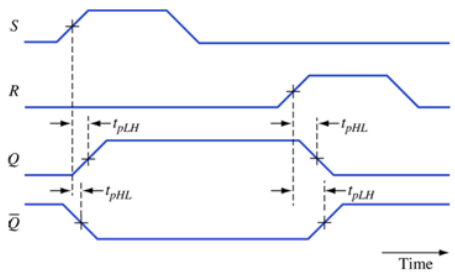


  - Latches & Progagation Delays

- Latches
  - Single-bit storage (memory)
  - Changes state at any time due to input change
  - Must guarantee a minimum pulse width to avoid metastability
  - Fast and cheap (small number of transistors)
  - Often used in high speed microprocessor design

Propagation delays in an *SR* latch.

Figure 6.7



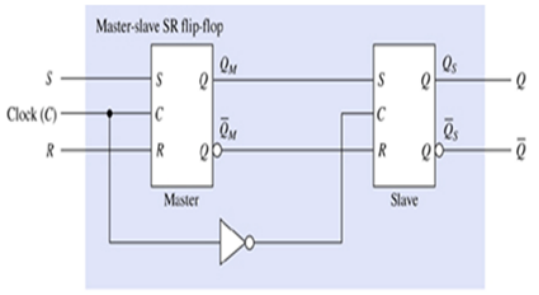
See the section 3.10.3 of chapter 3 for detailed descriptions of **Propagation delays** .

■ Master-Slave SR Flip-Flops (Pulse-Triggered Flip-Flops)

The Master-Slave SR Flip-Flop

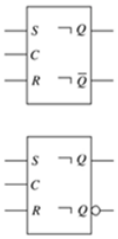
- Master-slave flip-flops
  - Also called pulse-triggered flip-flops
  - Also called edge-triggered flip-flops
- Without control signal (enable or clock), the output immediately responses when input changes, such as latch.
  - It is also called as transparency.
- If existing a control signal (enable or clock), the output will change with respect to control signal, such as flip-flop.

33



34

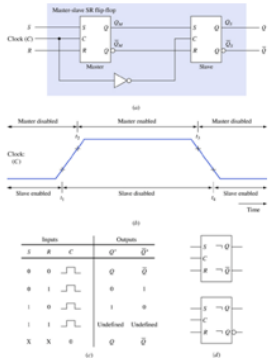
Inputs			Outputs	
S	R	C	Q^+	$\bar{Q}^+$
0	0	↑	Q	$\bar{Q}$
0	1	↑	0	1
1	0	↑	1	0
1	1	↑	Undefined	Undefined
X	X	0	Q	$\bar{Q}$



35

Master-slave *SR* flip-flop. (*a*) Logic diagram using gated *SR* latches. (*b*) Flip-flop action during the control signal. (*c*) Function table where Q^+ denotes the output Q in response to the inputs. (*c*) Two logic symbols.

Figure 6.12

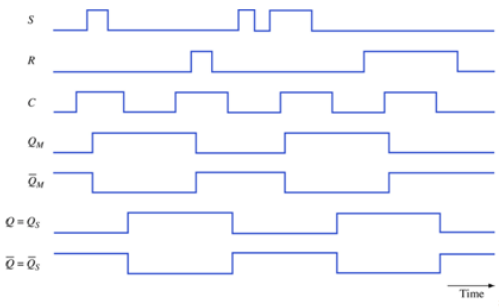


36

■ The Master-Slave SR Flip-Flop

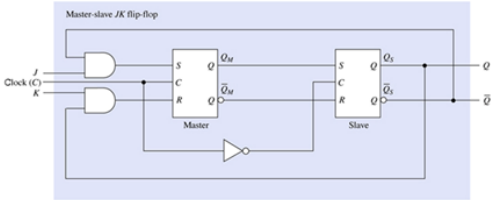
Timing diagram for a master-slave *SR* flip-flop.

Figure 6.13



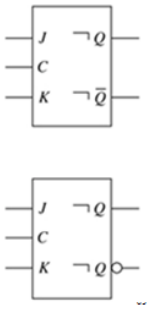
37

The Master-Slave JK Flip-Flop



38

Inputs			Outputs	
J	K	C	Q^+	$\overline{Q}^+$
0	0	⊓	Q	$\overline{Q}$
0	1	⊓	0	1
1	0	⊓	1	0
1	1	⊓	$\overline{Q}$	Q
X	X	0	Q	$\overline{Q}$

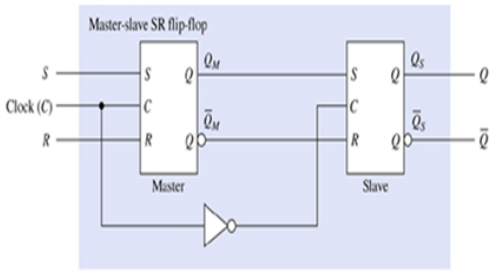


■ The Master-Slave JK Flip-Flop

The Master-Slave SR Flip-Flop

- Master-slave flip-flops
  - Also called pulse-triggered flip-flops
  - Also called edge-triggered flip-flops
- Without control signal (enable or clock), the output immediately responses when input changes, such as latch.
  - It is also called as transparency.
- If existing a control signal (enable or clock), the output will change with respect to control signal, such as flip-flop.

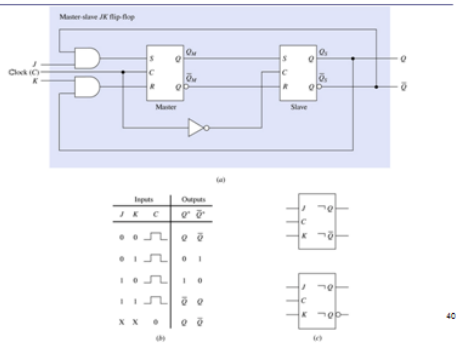
33



34

Master-slave *JK* flip-flop. (*a*) Logic diagram using gated *SR* latches. (*b*) Function table where Q^+ denotes the output Q in response to the inputs. (*c*) Two logic symbols.

Figure 6.14

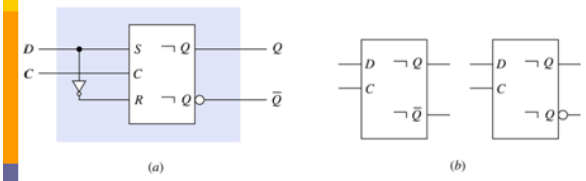


40

■ The Master-Slave D Flip-Flop

Master-slave *D* flip-flop. (*a*) Logic diagram using master-slave *SR* flip-flop (*b*) Two logic symbols.

Figure 6.16

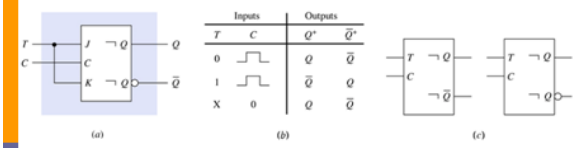


55

- The Master-Slave T Flip-Flop

Master-slave *T* flip-flop. (*a*) Logic diagram using a master-slave *JK* flip-flop. (*b*) Function table where Q^+ denotes the output *Q* in response to the inputs. (*c*) Two logic symbols.

Figure 6.17

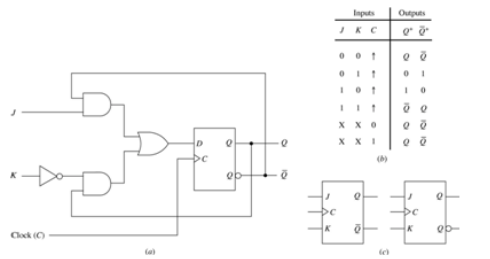


56

- Edge-Triggerred SR/JK/F/T Flip-Flop

Positive-edge-triggered *JK* flip-flop. (*a*) Logic diagram. (*b*) Function table where Q^+ denotes the output *Q* in response to the inputs. (*c*) Two logic symbols.

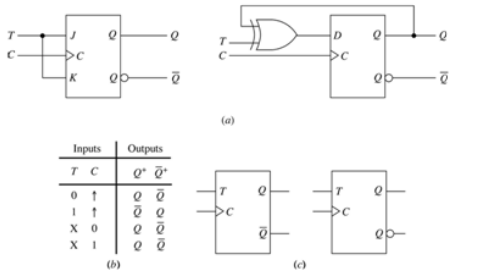
Figure 6.22



71

Positive-edge-triggered *T* flip-flop. (*a*) Logic diagram. (*b*) Function table where Q^+ denotes the output *Q* in response to the inputs. (*c*) Two logic symbols.

Figure 6.23



72

- Characteristic Equations

附件 3

Table 6.1 Simplified flip-flop function tables. Q denotes the current state and Q^+ denotes the resulting state as a consequence of the information inputs and the control signal. (*a*) *SR* flip-flop. (*b*) *D* flip-flop. (*c*) *JK* flip-flop. (*d*) *T* flip-flop.

S	R	Q^+
0	0	Q
0	1	0
1	0	1
1	1	–

(*a*)

D	Q^+
0	0
1	1

(*b*)

J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	$\bar{Q}$

(*c*)

T	Q^+
0	Q
1	$\bar{Q}$

(*d*)

77

Table 6.2 Flip-flop next-state tables. Q denotes the current state and Q^+ denotes the resulting state as a consequence of the information inputs and the control signal. (*a*) *SR* flip-flop. (*b*) *D* flip-flop. (*c*) *JK* flip-flop. (*d*) *T* flip-flop.

S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	–
1	1	1	–

Inputs not allowed

(*a*)

D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

(*b*)

J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

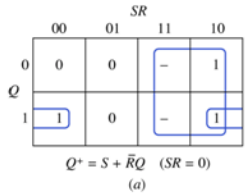
(*c*)

T	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

(*d*)

Characteristic equations. (*a*) Derivation of characteristic equation for an *SR* flip-flop. (*b*) Summary of characteristic equations.

Figure 6.25



(*a*)

Flip-flop type	Characteristic equation
SR	$Q^+ = S + \bar{R}Q \quad (SR = 0)$
JK	$Q^+ = J\bar{Q} + \bar{K}Q$
D	$Q^+ = D$
T	$Q^+ = T\bar{Q} + \bar{T}Q = T \oplus Q$

(*b*)

79

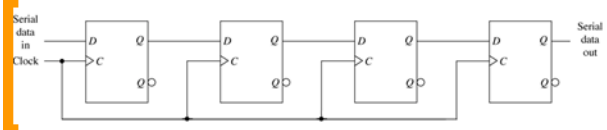
- Registers

- A register is a collection of flip-flops.
  - A register contains finite flip-flops
  - Each flip-flops stores 0 or 1.
  - The combination of flip-flops is known as state of content of the register
- Registers that are capable of moving information positionwise upon the occurrence of a clock signal is called *shift register*, such as
  - Serial-in, serial-out unidirectional shift register
  - Serial-in, parallel-out unidirectional shift register
  - Parallel-in unidirectional shift register

1

Serial-in, serial-out unidirectional shift register.

Figure 6.26



Let Q = 1011

If D = 0
Then Q = 0101

84

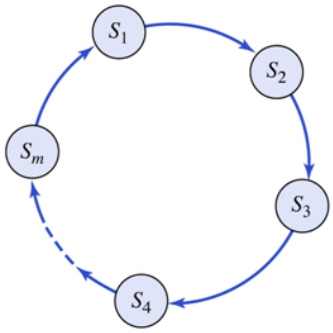
- Counters

- Counter is another example of a register which produces a specified output pattern sequence.
  - Is also called as pattern generator.
- The output pattern is a *state* of the counter.
  - The total number of states is called its *modulus*.
  - A counter has m distinct states, then it is called a *modulus-m counter* or *mod-m counter*.

91

State diagram of a counter.

Figure 6.30



92

伍、實施成效及影響（量化及質化）

實施成效:

1. 學員數: 102 人
2. 上課時數: 18 小時
3. 作業次數: 10 次作業
4. 數位電路範例: 10 種
5. 作業完成率: 782/1020=77%
6. 未完成率: 238/1020=23%
7. 硬體程式語: Schematic Layout
8. CAD 設計硬體: Altera Quartus II v9.1
9. 作業繳交明細表

附件 3

學員排序	學號	姓名	作業#1	作業#2	作業#3	作業#4	作業#5	作業#6	作業#7	作業#8	作業#9	作業#10
1	N/A	N/A	1	1	1	3	2	2	2	1	3	2
2	N/A	N/A	1	1	2	3	2	2	2	2	3	2
3	N/A	N/A	1	x	2	3	2	2	2	2	3	2
4	N/A	N/A	1	1	x	x	x	x	x	x	x	x
5	N/A	N/A	1	1	2	3	2	1	2	2	3	2
6	N/A	N/A	1	1	2	3	2	2	2	2	3	2
7	N/A	N/A	1	1	2	x	2	2	2	2	x	2
8	N/A	N/A	1	1	2	x	2	2	x	2	x	2
9	N/A	N/A	1	x	x	x	x	x	x	x	x	x
10	N/A	N/A	1	x	x	x	x	x	x	x	x	x
11	N/A	N/A	x	1	x	x	x	x	x	x	x	x
12	N/A	N/A	1	1	1	3	2	x	2	1	3	2
13	N/A	N/A	1	1	x	3	x	1	x	x	3	x
14	N/A	N/A	1	1	1	3	2	x	1	1	3	2
15	N/A	N/A	1	1	x	x	x	x	x	x	x	x
16	N/A	N/A	1	1	x	x	x	x	x	x	x	x
17	N/A	N/A	1	1	2	3	2	2	2	2	3	2
18	N/A	N/A	1	1	x	3	2	1	1	1	3	2
19	N/A	N/A	1	1	1	1	2	x	x	1	1	2
20	N/A	N/A	1	1	2	x	x	x	x	2	x	x
21	N/A	N/A	1	1	x	x	x	x	x	x	x	x
22	N/A	N/A	1	1	x	3	2	x	x	x	3	2
23	N/A	N/A	1	1	x	1	x	x	x	x	1	x
24	N/A	N/A	1	x	x	3	2	2	x	x	3	2
25	N/A	N/A	1	1	2	3	2	2	2	2	3	2
26	N/A	N/A	1	x	x	x	x	x	x	x	x	x
27	N/A	N/A	1	1	2	3	x	x	x	2	3	x
28	N/A	N/A	1	1	1	x	1	x	x	1	x	1
29	N/A	N/A	1	1	2	x	x	x	x	2	x	x
30	N/A	N/A	x	x	x	x	x	x	x	x	x	x
31	N/A	N/A	1	1	1	1	1	x	x	1	1	1

32	N/A	N/A	1	1	x	x	x	x	x	x	x	x
33	N/A	N/A	1	1	x	x	x	x	x	x	x	x
34	N/A	N/A	1	1	x	3	2	2	1	1	3	2
35	N/A	N/A	1	1	2	2	2	2	2	2	2	2
36	N/A	N/A	1	1	2	3	2	2	x	2	3	2
37	N/A	N/A	1	1	1	1	2	2	2	1	1	2
38	N/A	N/A	1	1	x	3	x	x	x	x	3	x
39	N/A	N/A	1	1	1	x	x	x	x	1	x	x
40	N/A	N/A	1	1	x	x	x	x	x	x	x	x
41	N/A	N/A	1	1	2	3	2	1	2	2	3	2
42	N/A	N/A	1	x	2	x	x	1	x	2	x	x
43	N/A	N/A	1	1	2	3	x	1	x	2	3	x
44	N/A	N/A	1	x	x	3	x	x	x	x	3	x
45	N/A	N/A	1	1	2	3	2	2	x	2	3	2
46	N/A	N/A	1	x	x	x	x	x	x	x	x	x
47	N/A	N/A	x	x	x	x	x	x	x	x	x	x
48	N/A	N/A	1	1	1	3	x	x	x	1	3	x
49	N/A	N/A	1	1	x	3	2	x	x	x	3	2
50	N/A	N/A	x	x	1	1	1	x	1	1	1	1
51	N/A	N/A	1	1	2	3	x	x	2	2	3	x
52	N/A	N/A	1	1	2	3	2	2	2	2	3	2
53	N/A	N/A	1	1	2	3	2	2	2	2	3	2
54	N/A	N/A	1	1	1	2	2	2	2	1	2	2
55	N/A	N/A	1	1	2	x	2	x	x	2	x	2
56	N/A	N/A	1	1	2	1	x	1	2	2	1	x
57	N/A	N/A	1	1	2	3	2	2	2	2	3	2
58	N/A	N/A	1	1	1	1	x	1	x	1	1	x
59	N/A	N/A	1	1	1	x	x	x	2	1	x	x
60	N/A	N/A	1	1	2	3	2	x	x	2	3	2
61	N/A	N/A	x	1	2	x	2	x	1	2	x	2
62	N/A	N/A	1	1	2	3	2	2	2	2	3	2
63	N/A	N/A	1	1	2	3	2	2	1	2	3	2
64	N/A	N/A	1	1	2	3	2	2	1	2	3	2
65	N/A	N/A	1	1	1	x	2	2	x	1	2	2
66	N/A	N/A	1	1	2	3	2	x	x	2	3	2
67	N/A	N/A	1	1	1	3	2	2	2	1	3	2
68	N/A	N/A	1	1	2	x	x	x	2	2	x	x

69	N/A	N/A	1	1	2	x	2	2	x	2	x	2
70	N/A	N/A	1	x	x	x	x	x	x	x	x	x
71	N/A	N/A	1	1	2	3	2	2	2	2	3	2
72	N/A	N/A	x	x	x	x	x	x	x	x	x	x
73	N/A	N/A	1	x	2	x	2	2	x	2	x	2
74	N/A	N/A	x	x	x	3	2	1	x	1	3	2
75	N/A	N/A	1	1	1	3	2	2	2	1	3	2
76	N/A	N/A	1	1	2	3	2	2	2	2	3	2
77	N/A	N/A	1	1	1	1	1	x	x	1	1	1
78	N/A	N/A	1	1	1	3	x	2	x	1	3	x
79	N/A	N/A	1	1	x	3	x	2	2	x	3	x
80	N/A	N/A	1	1	2	x	x	x	x	2	x	x
81	N/A	N/A	1	1	2	3	x	2	x	2	3	x
82	N/A	N/A	1	1	2	3	2	2	x	2	3	2
83	N/A	N/A	1	1	1	3	x	2	x	1	3	x
84	N/A	N/A	1	1	1	3	x	2	x	1	3	x
85	N/A	N/A	1	1	1	3	2	2	x	1	3	2
86	N/A	N/A	1	1	1	1	2	2	x	1	1	2
87	N/A	N/A	x	1	2	3	2	1	x	2	3	2
88	N/A	N/A	1	1	1	3	2	2	x	1	3	2
89	N/A	N/A	x	1	1	x	x	x	x	1	x	x
90	N/A	N/A	x	1	1	1	1	1	2	1	1	1
91	N/A	N/A	x	x	x	x	x	x	x	x	x	x
92	N/A	N/A	x	x	1	x	x	x	x	1	x	x
93	N/A	N/A	1	x	2	3	x	2	2	2	3	x
94	N/A	N/A	1	x	2	3	2	2	2	2	3	2
95	N/A	N/A	x	1	2	3	x	2	2	2	3	x
96	N/A	N/A	1	1	2	3	2	2	2	2	3	2
97	N/A	N/A	x	1	1	3	x	2	2	1	3	x
98	N/A	N/A	1	1	2	x	2	2	x	2	x	2
99	N/A	N/A	x	1	x	1	1	1	1	1	1	1
100	N/A	N/A	x	x	1	x	1	2	x	1	x	1
101	N/A	N/A	1	1	2	3	x	2	x	2	3	x
102	N/A	N/A	x	x	2	x	2	2	1	2	1	2

實施影響:

對組合電路與序列電路的了解與實作有實際操作的機會，讓學員設計數位基本元件，同時亦了解設計時程式碼的編排，並觀察軟體設計與硬體電路的連繫。為讓學員對此部分有更深刻的理解，本教材特別對數位電路模擬(simulation)提出許多的範例與說明，包括了大量的時序圖(timing diagram)，寄望由這些例證，能了解數位系統設計的基本原理，進而發展硬體設計的能力，提供學員在數位領域的競爭力，進入職場後能更快速地融入工作，追求自我生涯的更高層境界。

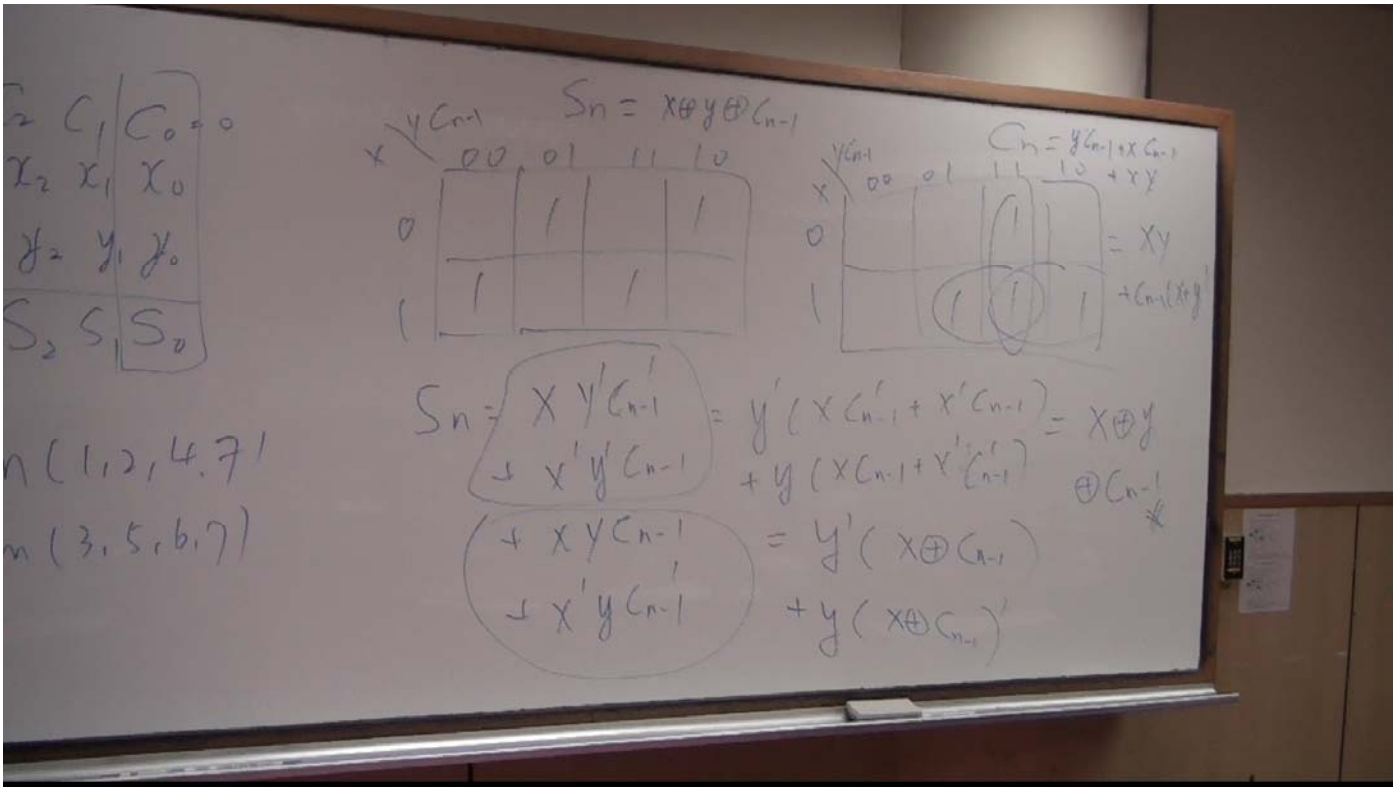
陸、結論

組合電路與序列電路對數位晶片與數位電路的了解與實作有很大的助益。透過有系統的教材資料，針對各種不同的數位電路元件，解說相對應的實際範例，可以讓學員同時複習元件的基本運作原理，同時亦了解設計時數位電路的基本原理，並同時可觀察瞭解軟硬體的設計與其間的連接。對大部份的初學者，數位訊號如何傳遞與在不同時間(timing)時訊號如何改變，更對於訊號間如何互相影響，無法立刻理解。為讓學員對此部分有更深刻的理解，本教材特別對數位電路模擬(simulation)提出許多的範例與說明，包括了大量的時序圖(timing diagram)，寄望由這些例證，能了解組合電路與序列電路的基本原則，進而發展硬體設計的能力，提供學員在數位領域的競爭力，進入職場後能更快

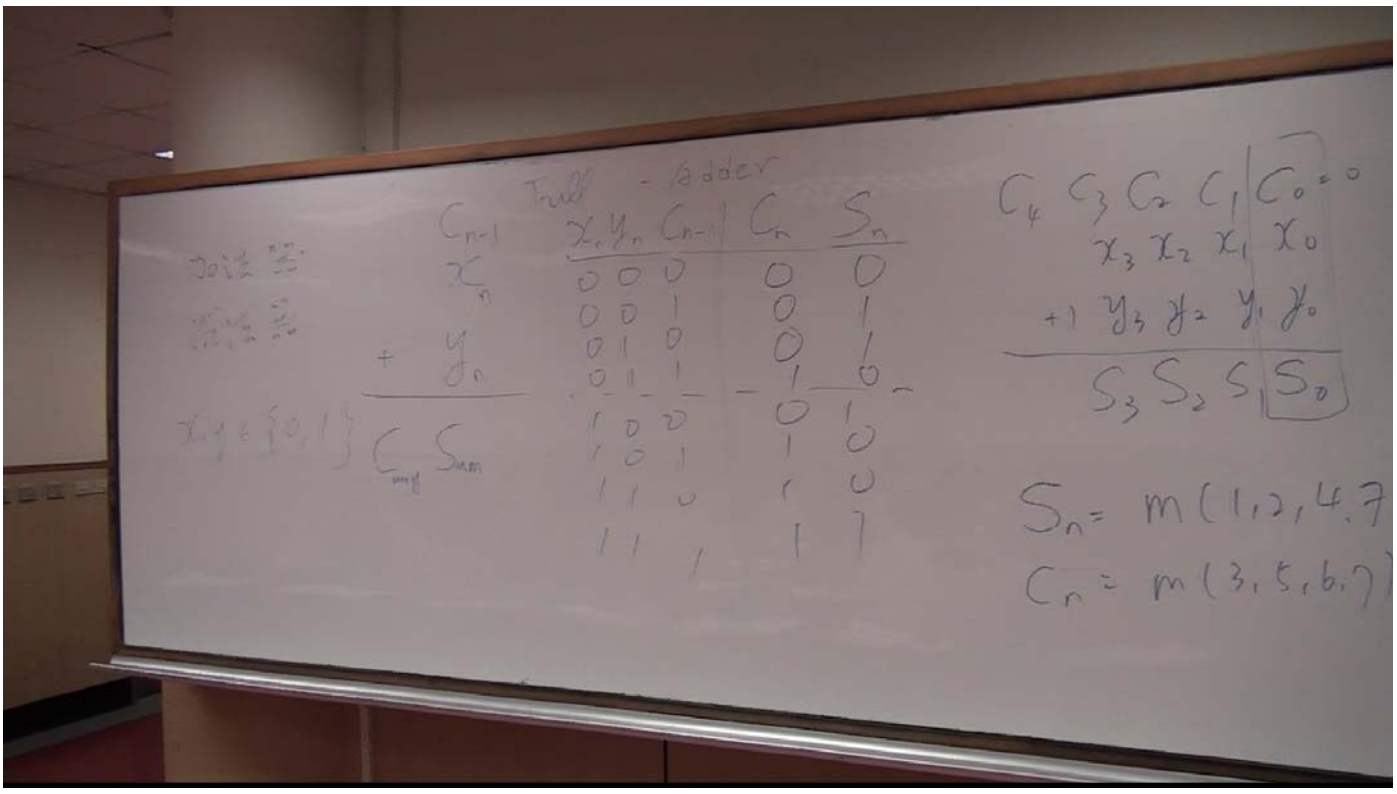
速地融入工作，追求自我生涯的更高層境界。

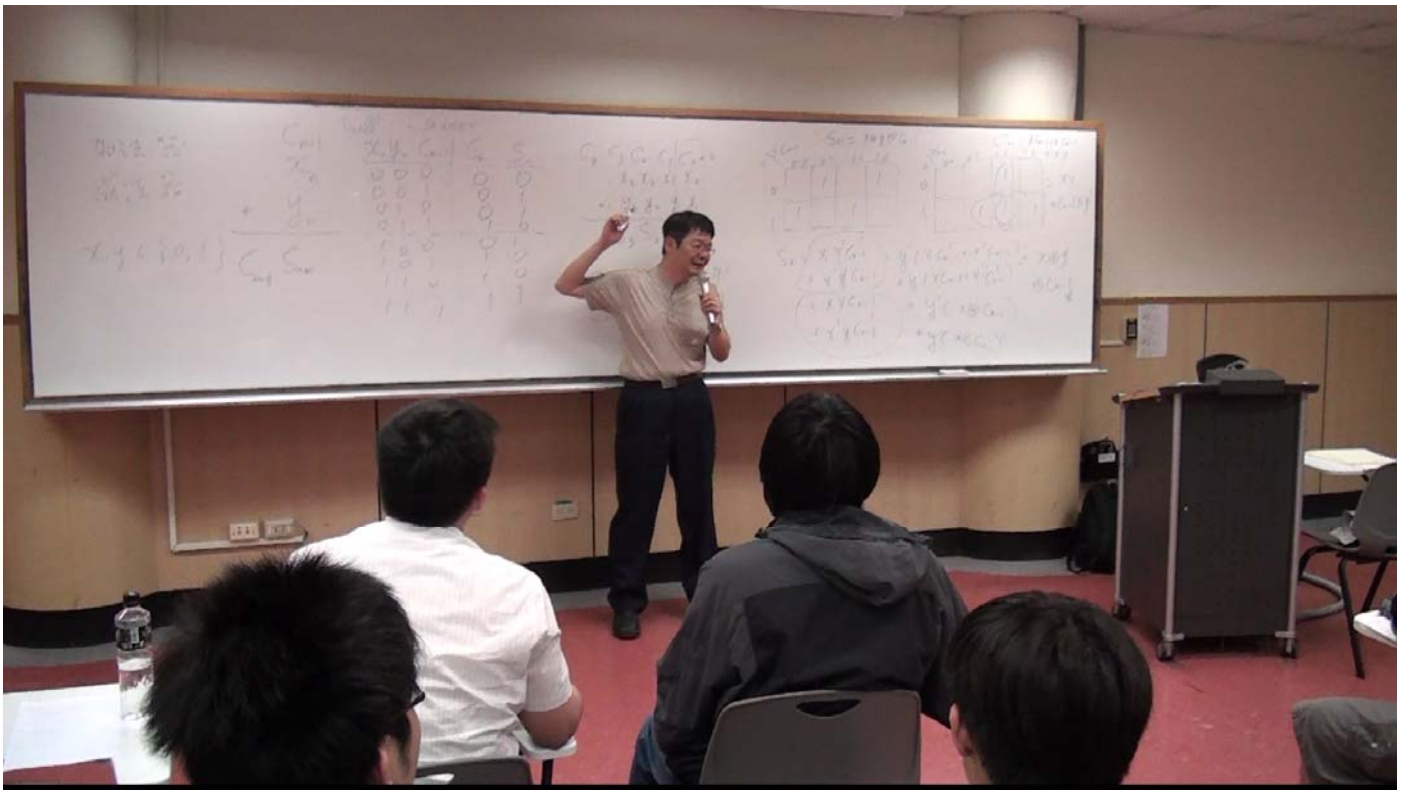
柒、執行計畫活動照片

附件 3

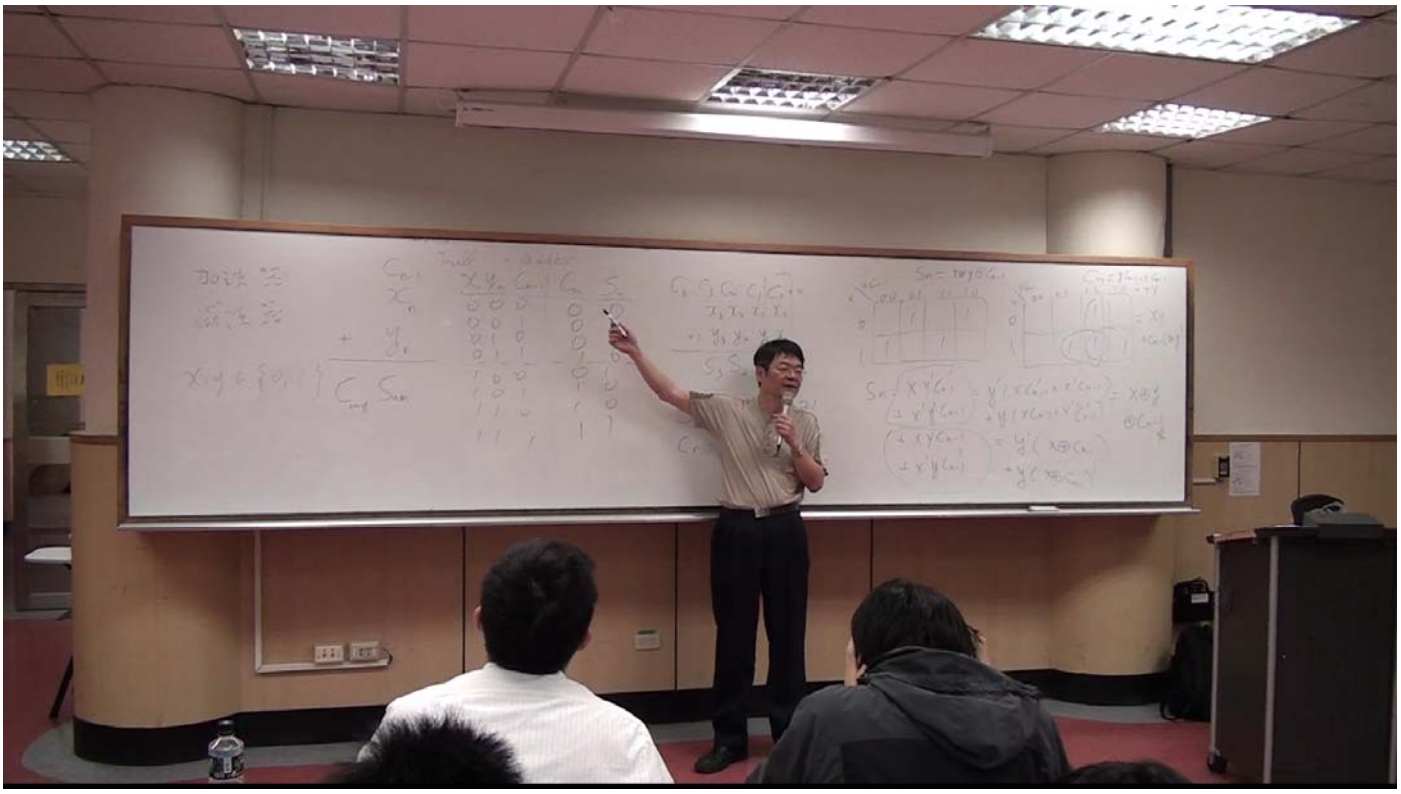

Sn = X⊕y⊕Cn-1
Cn = yCn-1 + xCn-1 + xy
Sn = X y' Cn-1 + x' y' Cn-1 + x y Cn-1 + x' y Cn-1
= y' (x Cn-1 + x' Cn-1) + y (x Cn-1 + x' Cn-1)'
= y' (X⊕Cn-1) + y (X⊕Cn-1)'
= X⊕y ⊕Cn-1

附件 3





附件 3



捌、附件

- 光碟片

備註：

1. 本報告書大綱得視需要自行增列項目。

2. 成果報告書須另以光碟儲存，並附加執行計畫活動照片電子檔。